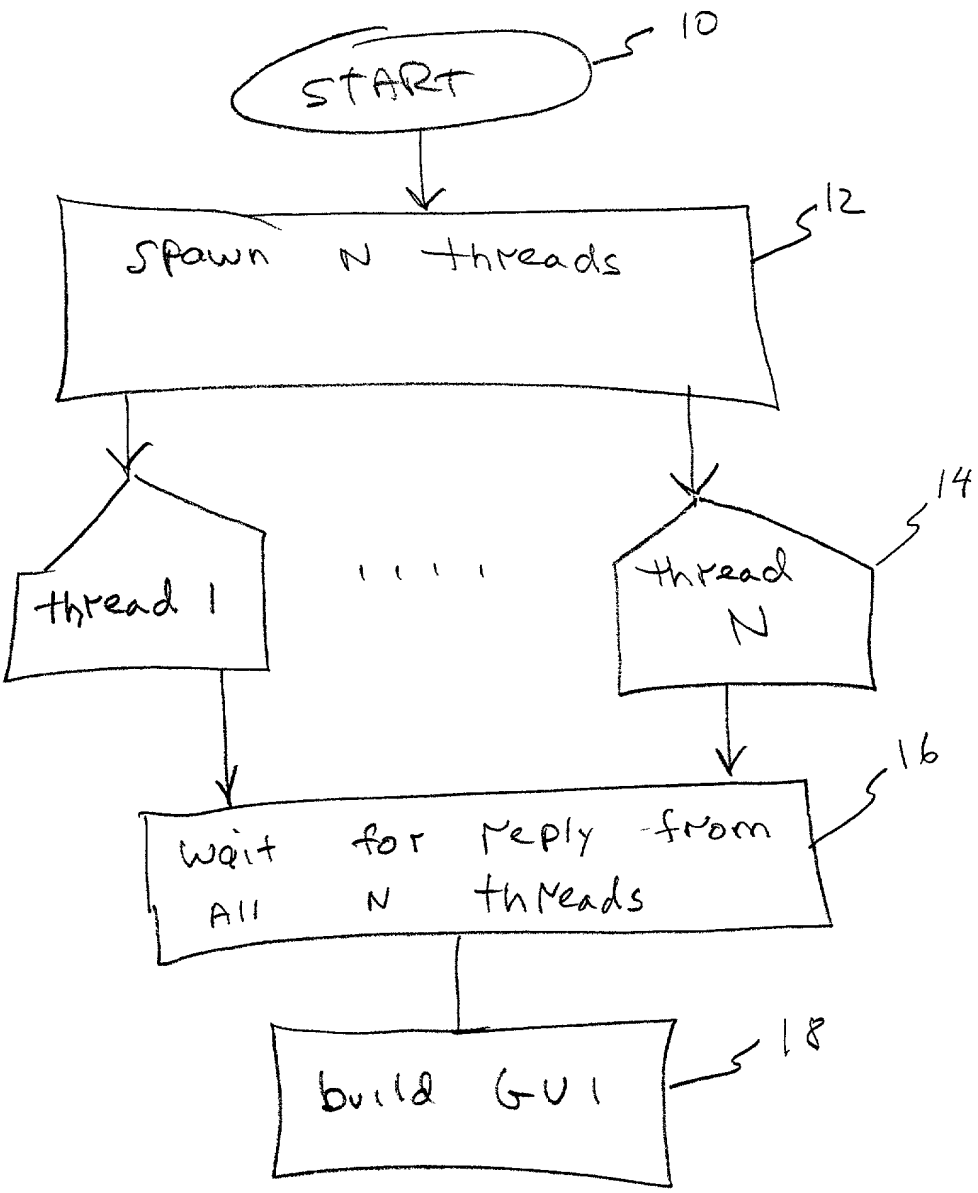

Fig. 1 (prior art)

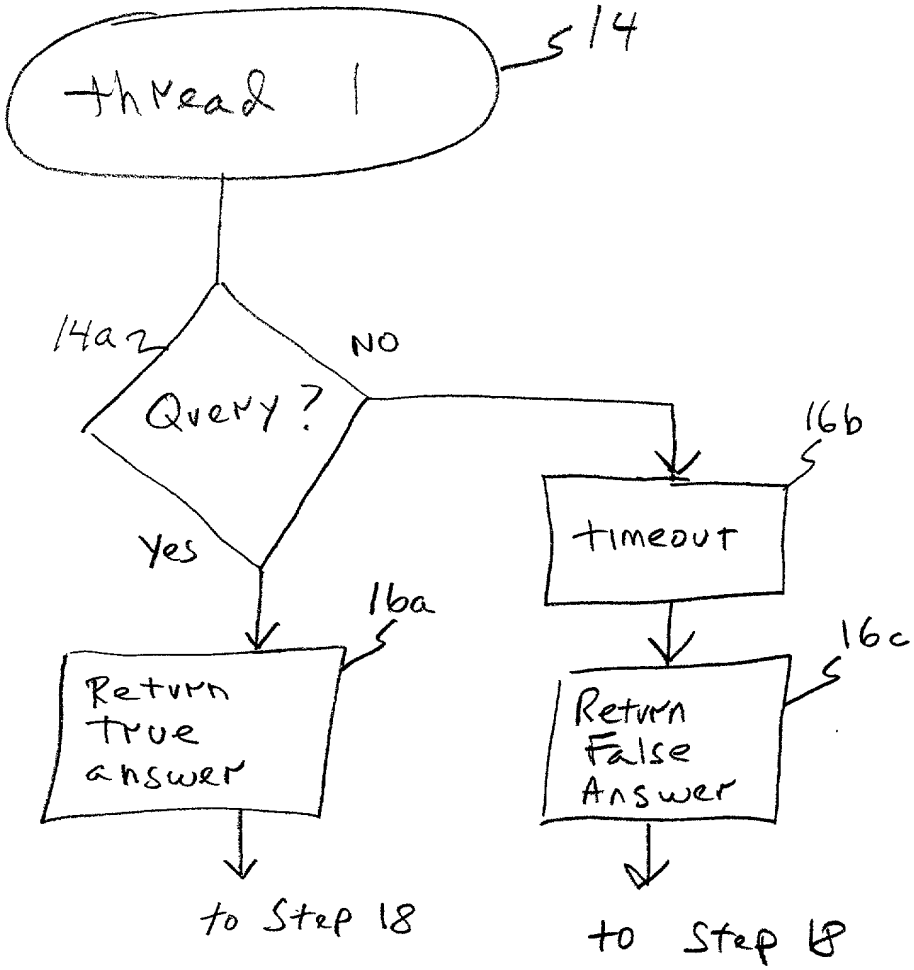

Fig. 2 (prior art)

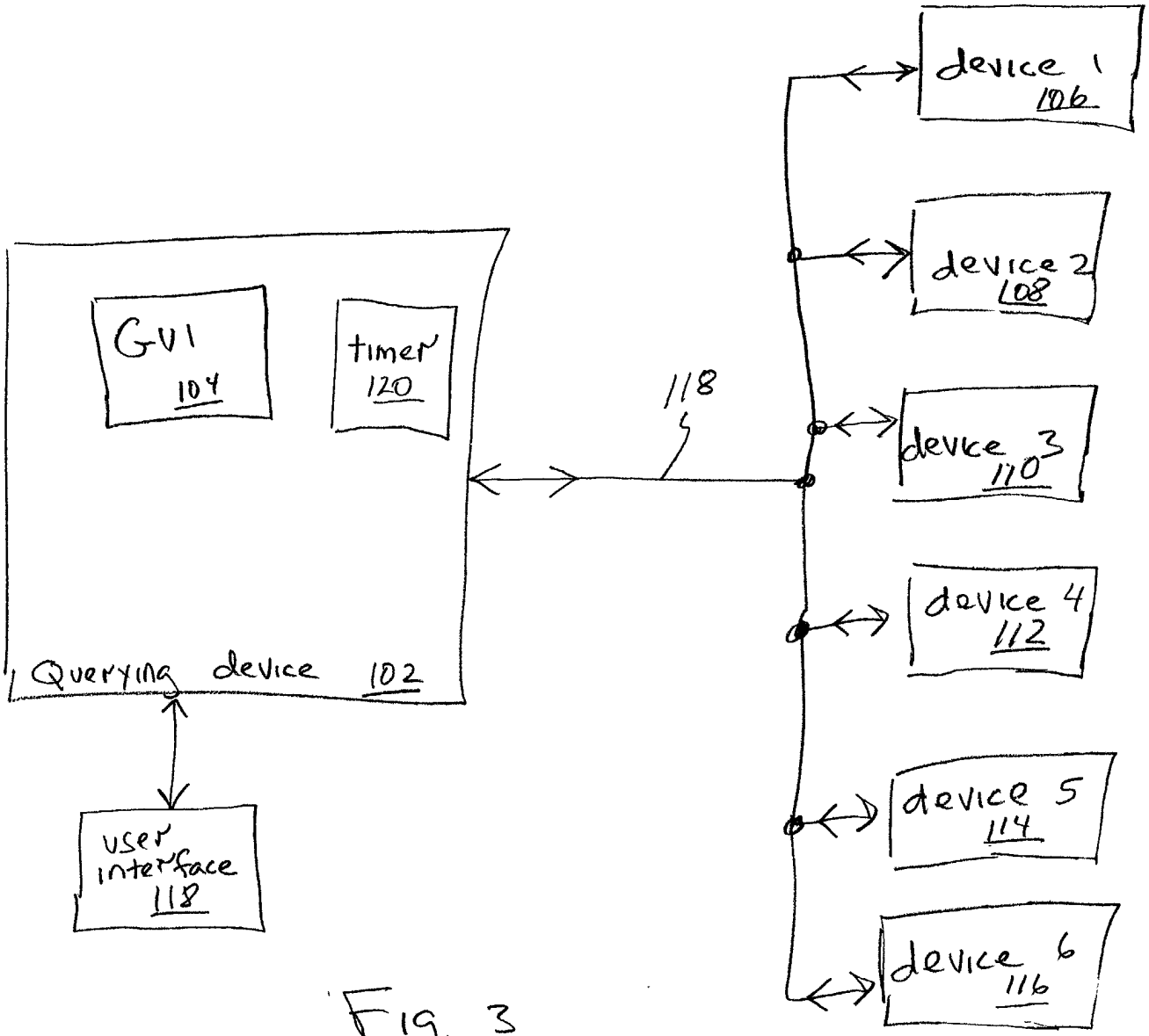

Fig. 3

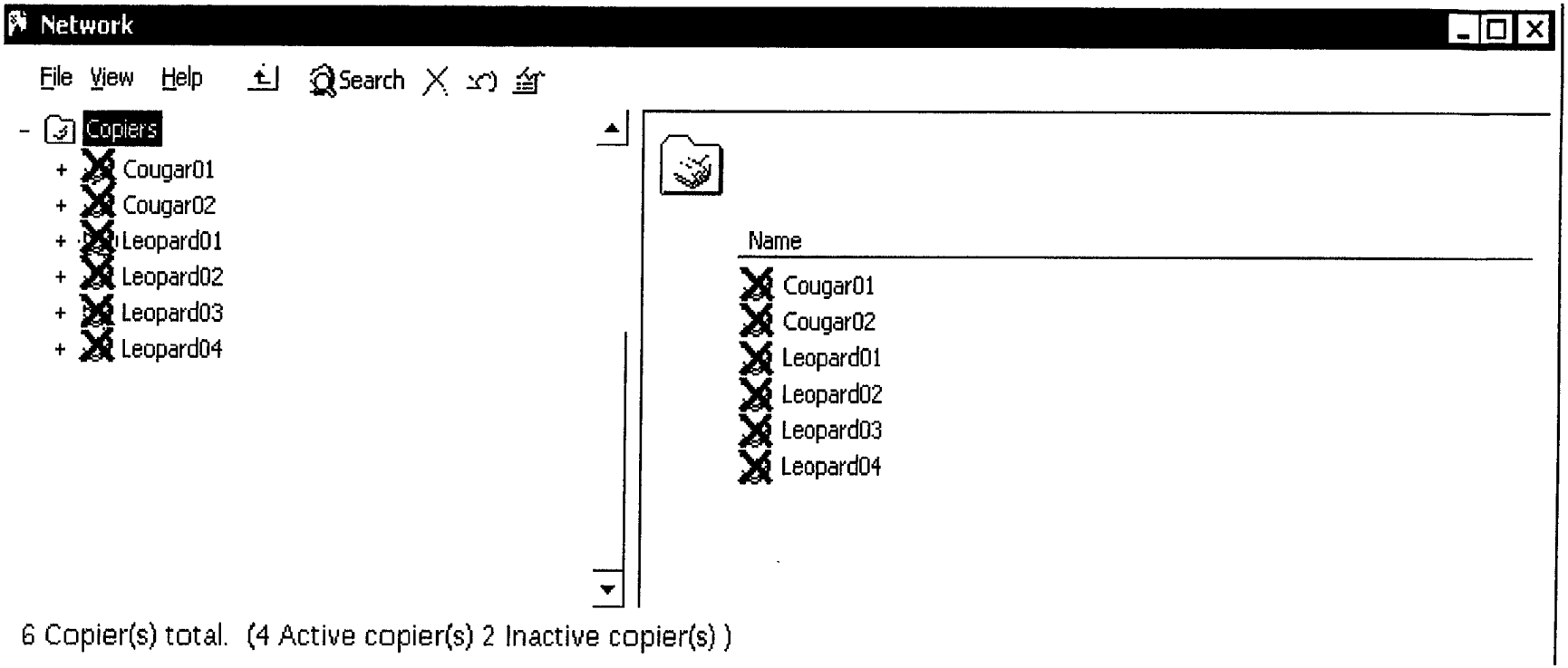
Network
File View Help
Search
Copiers
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
Name
Cougar01
Cougar02
Leopard01
Leopard02
Leopard03
Leopard04
6 Copier(s) total. (4 Active copier(s) 2 Inactive copier(s))

Fig. 4

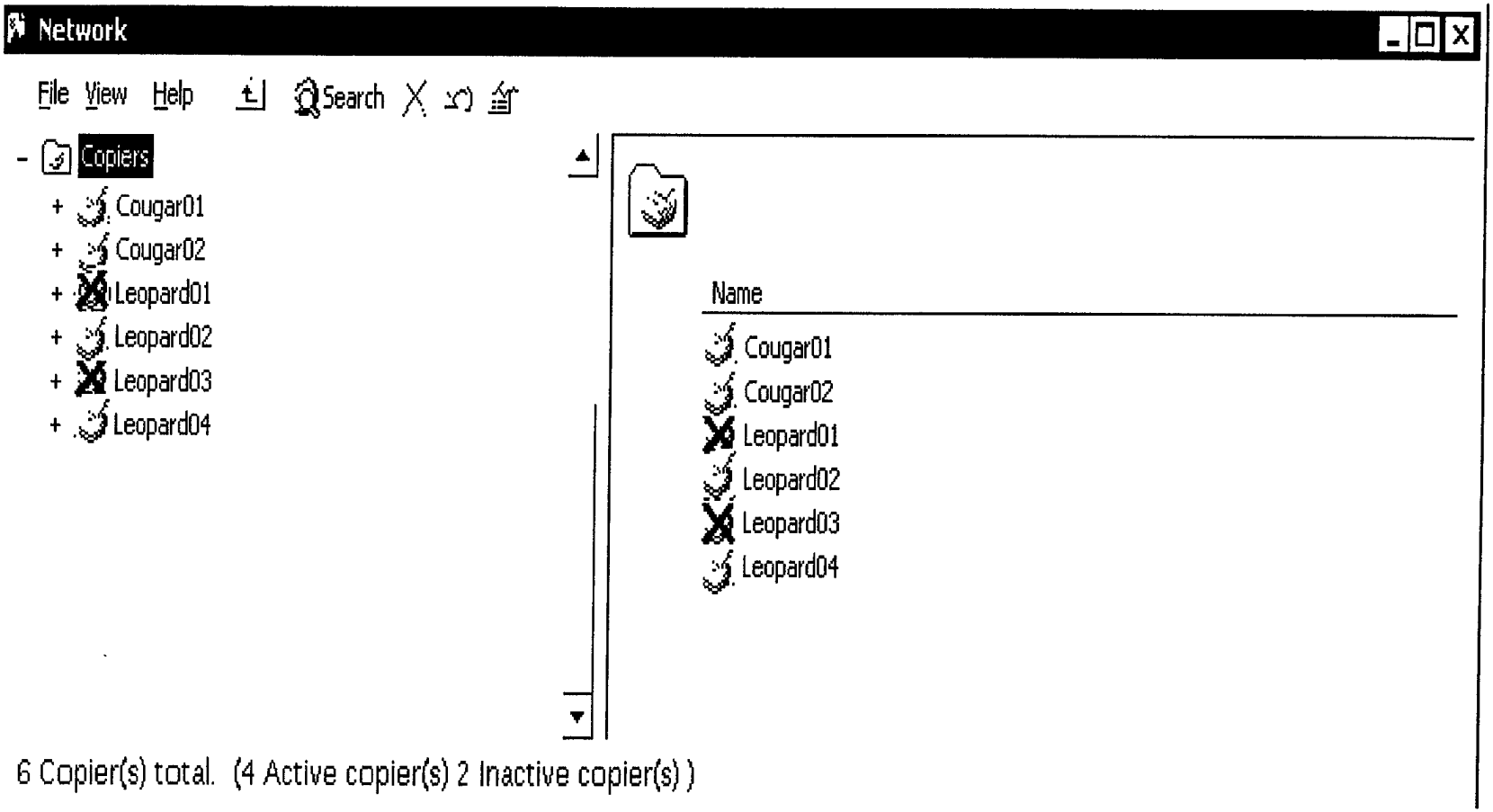

Fig. 5

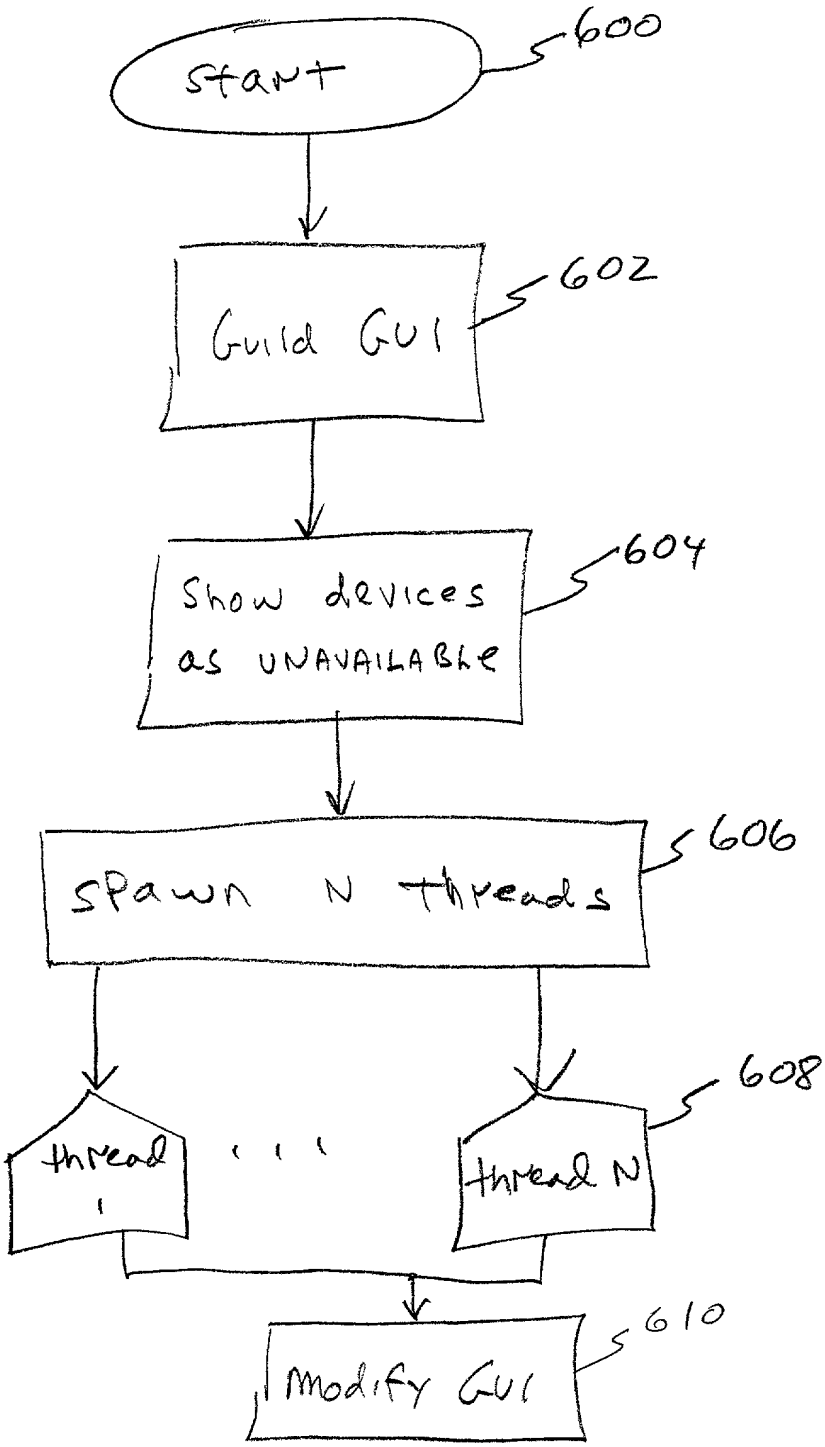
start
600
Build GUI
602
Show devices as UNAVAILABLE
604
spawn N threads
606
thread 1
. . .
thread N
608
Modify GUI
610

Fig. 6

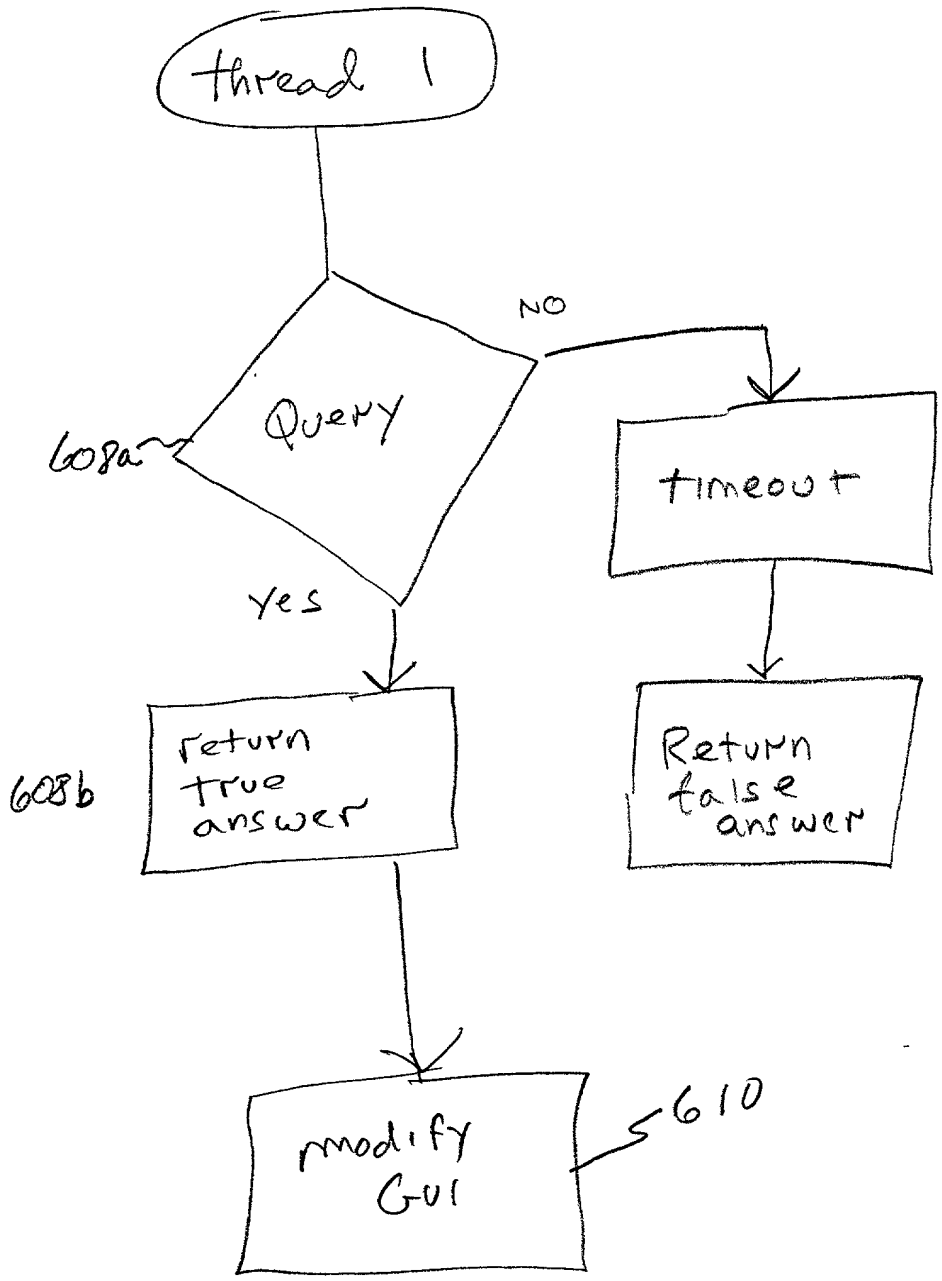
thread 1
Query
608a
NO
timeout
yes
608b
return true answer
Return false answer
modify GUI
610

Fig. 7

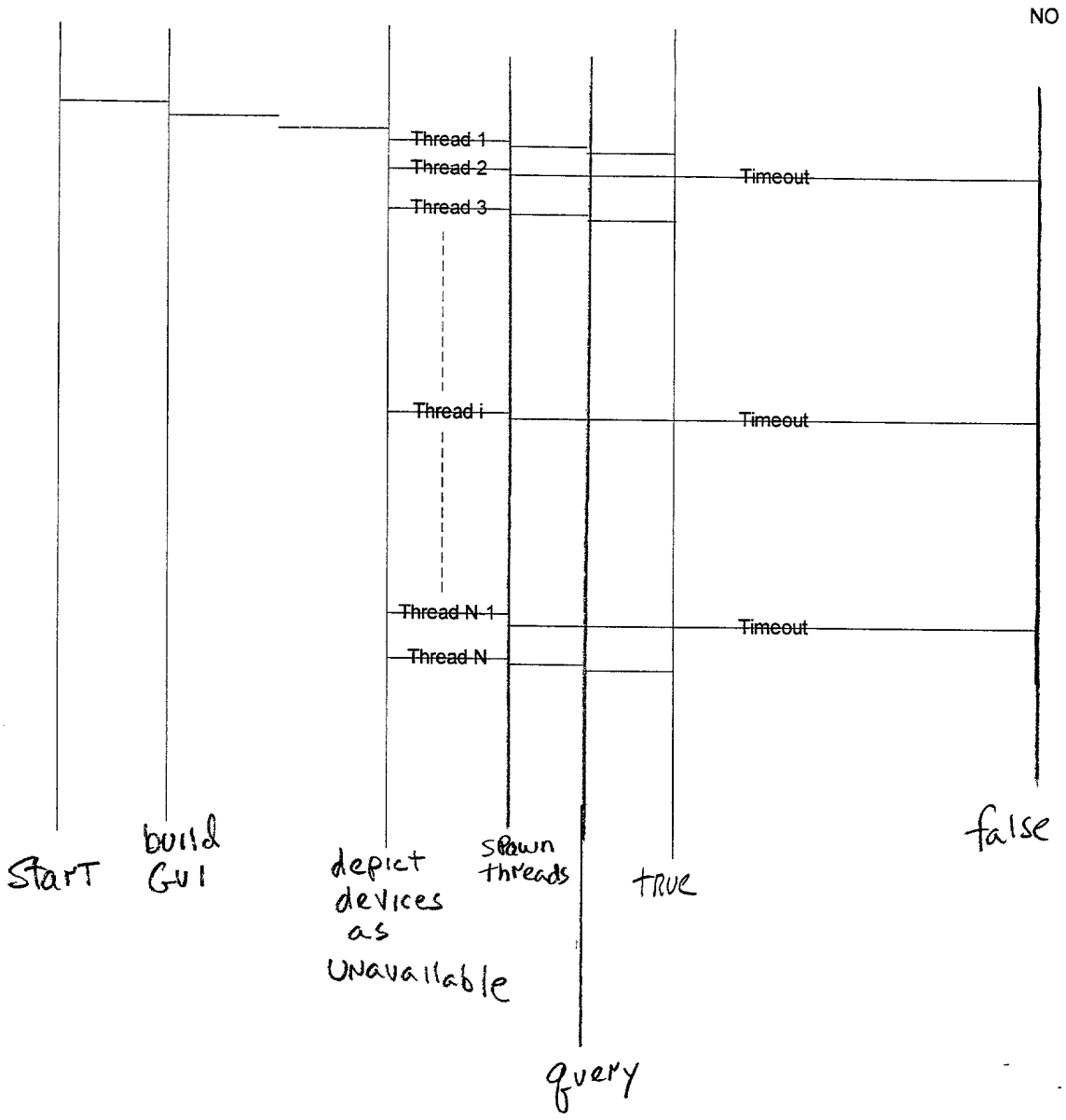
NO
Thread 1
Thread 2
Thread 3
Timeout
Thread i
Timeout
Thread N-1
Timeout
Thread N
Start
build GUI
depict devices as Unavailable
spawn threads
TRUE
false
query

Fig. 8

Code:

```
// DoEnable() gets an array of GUI components, and will enable/disable the components based on their availability in the network.
void DoEnable(GUIComponent arGUI[], unsigned short  sz)
{
        // Disable entire GUI
        for (unsigned short I=0;I<sz;I++)
                arGUI[I].DisableComponent ();

        for (I=0;I<sz;I++)
        {
                AfxBeginThread(ValidateRemoteHost, (LPVOID) &arGUI[I]);
        }
}
```

Fig. 9

```
struct GUIComponent
{
        // will return IP address associated with the network component that the current GUI element represents.
        unsigned long GetIPAddress ();
        void EnableComponent(); // display component as active
        void DisableComponent(); // display component as inactive.
};

// worker thread which will attempt a socket connection with remote host.
UINT ValidateRemoteHost( LPVOID pParam )
{
        GUIComponent * pGUIComponent =( GUIComponent *)pParam;

        if (QueryRemoteHost(pGUIComponent->GetIPAddress()) )
                pGUIComponent->EnableComponent();
        return 0;
}
```

Fig. 10

```
BOOL QueryRemoteHost(unsigned long lIpAdd)
{
	BOOL bRetVal=FALSE;
	// create new socket
	SOCKET S = socket( AF_INET, SOCK_STREAM, IPPROTO_IP);
	if ( INVALID_SOCKET == S)
		return FALSE;

	SOCKADDR_IN dest_sin;
	ZeroMemory(&dest_sin, sizeof(SOCKADDR_IN));
	dest_sin.sin_family = AF_INET;
	dest_sin.sin_port = htons(21);
	dest_sin.sin_addr.S_un.S_addr = lIpAdd;

	if (SOCKET_ERROR == connect(S,(PSOCKADDR)&dest_sin, sizeof( dest_sin)))
		bRetVal = FALSE;
	else
		bRetVal = TRUE;

	closesocket(S);
	return bRetVal;
}
```

Fig. 11

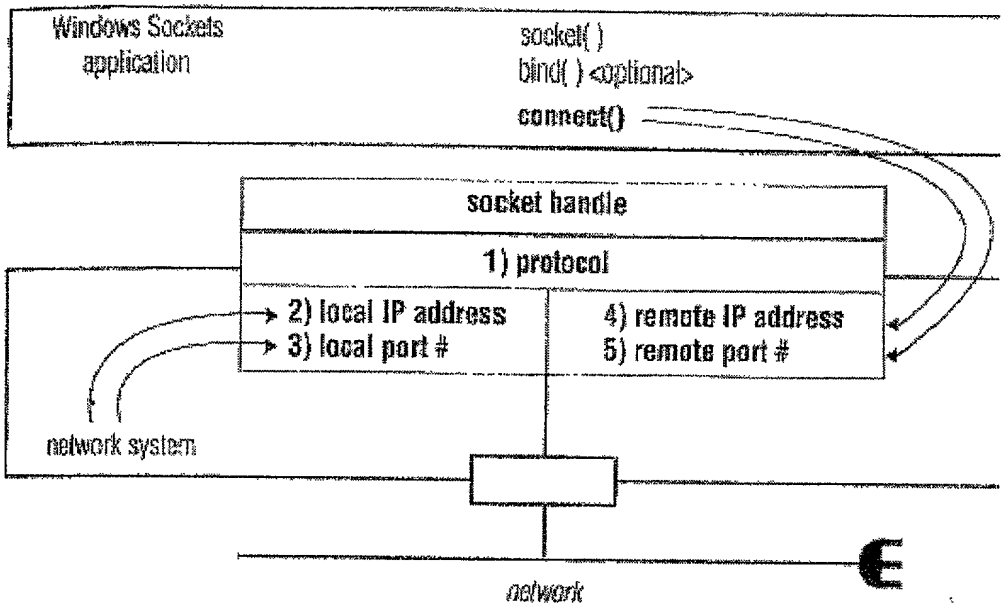
Windows Sockets application
socket( )
bind( ) <optional>
connect()
socket handle
1) protocol
2) local IP address
3) local port #
4) remote IP address
5) remote port #
network system
network

Fig. 12

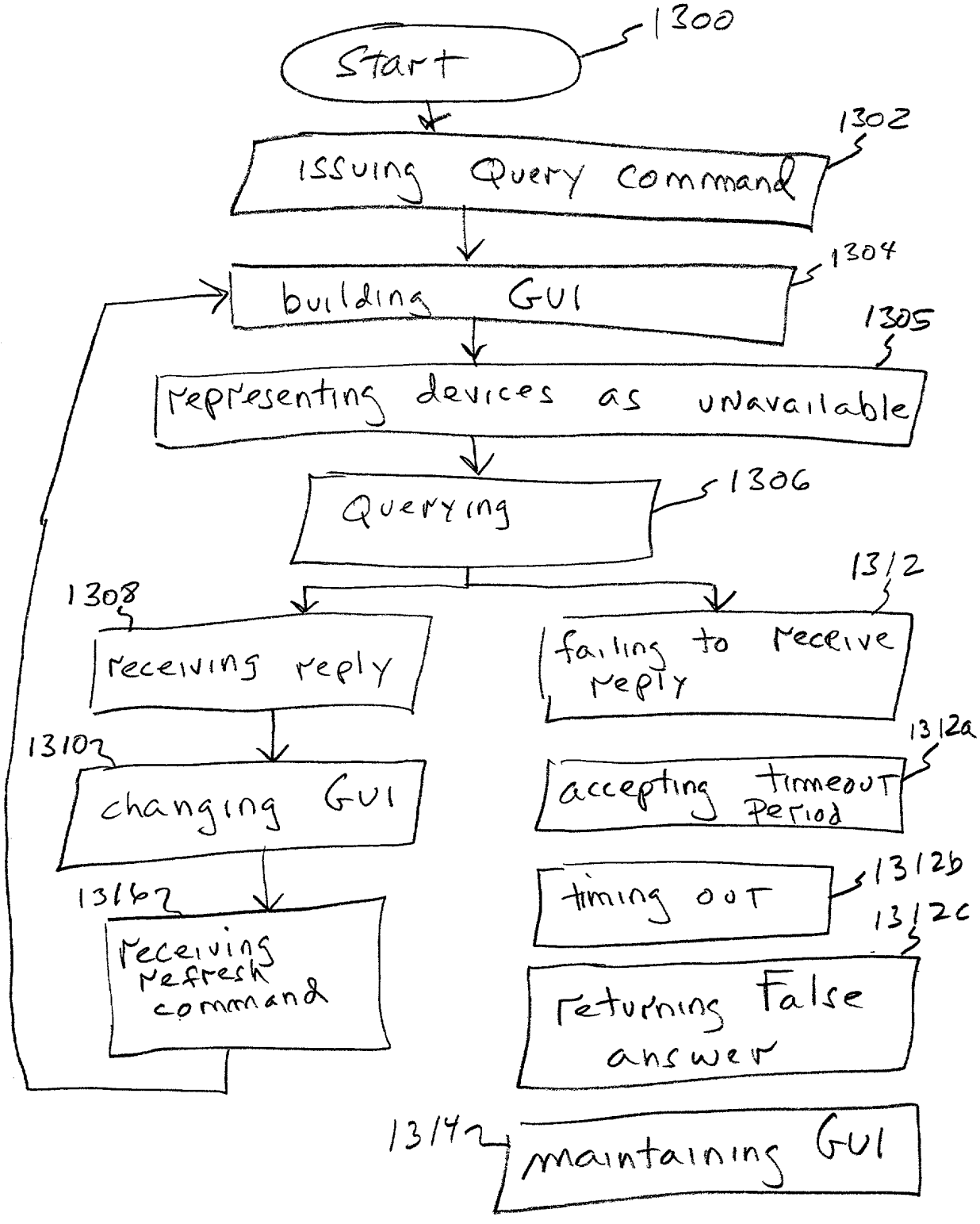
Start
1300
issuing Query command
1302
building GUI
1304
representing devices as unavailable
1305
Querying
1306
1308
receiving reply
13102
changing GUI
13162
receiving refresh command
1312
failing to receive reply
1312a
accepting timeout period
1312b
timing out
1312c
returning False answer
13142
maintaining GUI

Fig. 13

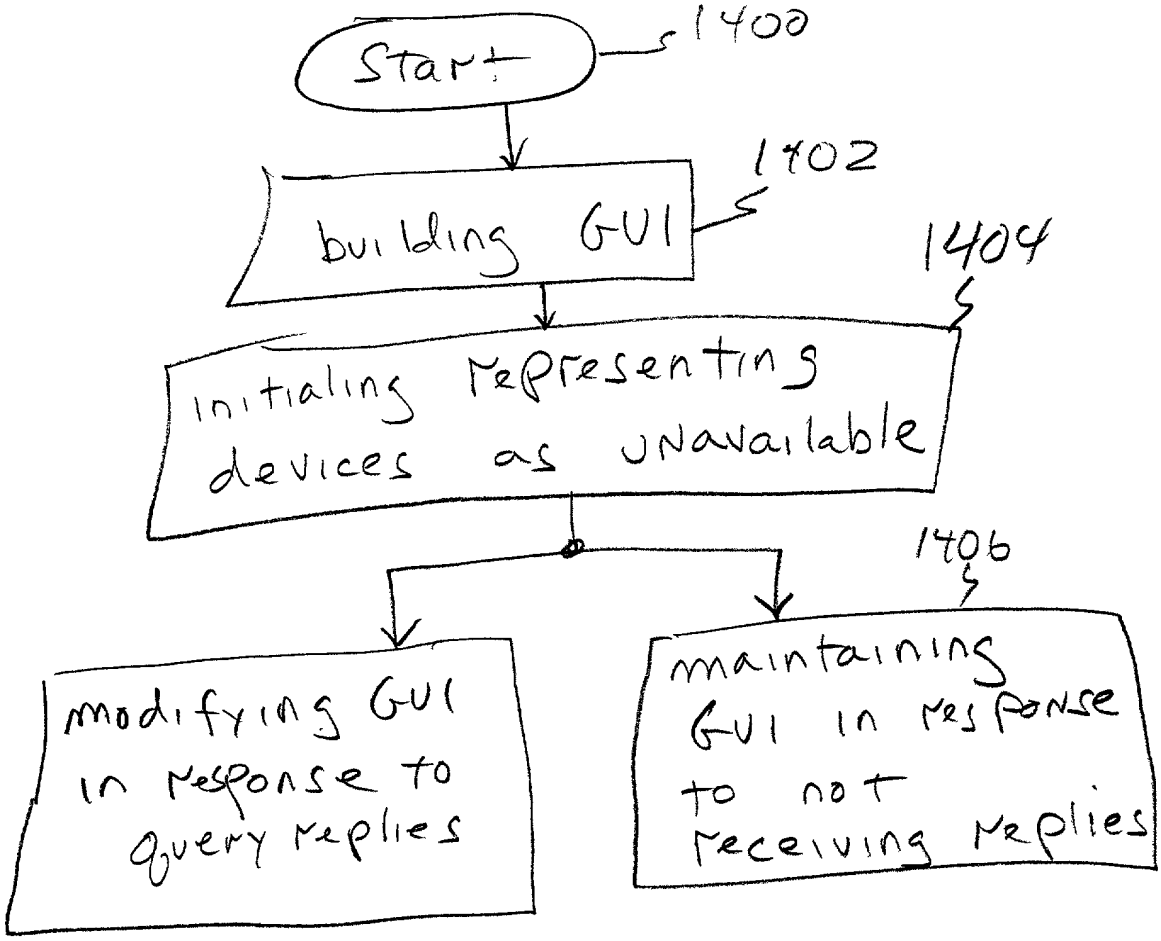
Start
1400
building GUI
1402
initialing representing devices as unavailable
1404
modifying GUI in response to query replies
maintaining GUI in response to not receiving replies
1406

Fig. 14